Amendments to the Claims

This listing of claims will replace all prior versions, and listings, of claims in the application.

1. (Currently Amended) A method of managing a lock utilized by a plurality of threads executing on a computing device to coordinate access to a shared resource, the method comprising:

selecting by one of the threads, an action to be performed by the thread upon the lock, wherein the action is selected from a group comprising:

acquiring the lock,

trying to acquire the lock, and

releasing the lock;

asynchronously querying and receiving a ~~current~~ first state of the lock as a current state of the lock by the thread, the lock being considered to be in any one of at least four states in any point in time, the states defined by a state machine associated with the lock ~~and each state is represented by a multi-part state value, each multi-part state value including a flag value, a first thread value, and a last thread value~~;

speculatively determining by the thread, ~~the~~ a ~~next~~ second state of the lock~~, where the next state is the state of the lock if the thread proceeds to perform the selected action and the thread is successful~~ based at least in part on the first state and the selected action; and

attempting to perform by the thread, the selected action to transition the lock from the ~~current~~ first state to the speculatively determined ~~next~~second state, the attempting including determining if the first state remains the current state of the lock, and, if the first state remains the current state, performing the selected action to transition the current state of the lock to the second state.

2. (Currently Amended) The method of claim 1, further including, if the state transition fails and if the selected action was either acquiring or releasing the lock, repeating, until the state transition succeeds:

asynchronously querying the ~~current~~ first state of the lock;

speculatively determining the ~~next state~~second state of the lock;

attempting to transition the lock from the queried ~~current~~ first state to the speculatively determined ~~next~~ second state.

3. (Currently Amended) The method of claim 2, further including, if

the state transition succeeds,

the selected action is acquiring the lock, and

the speculatively determined ~~next~~second state represents the acquisition of the lock,

indicating the acquisition of the lock.

4. (Currently Amended) The method of claim 3, further including, if

the state transition succeeds,

the selected action is acquiring the lock, and

the speculatively determined ~~next~~second state does not represent the acquisition of the lock,

adding the thread to the end of a queue of threads waiting to acquire the lock;

waiting to receive notification that the thread may acquire the lock; and

indicating the acquisition of the lock.

5. (Currently Amended) The method of claim 2, further including, if

the state transition succeeds, and

the selected action is releasing the lock,

determining the number of threads in a queue to acquire the lock utilizing the speculatively determined ~~next~~second state of the lock.

6. (Original) The method of claim 5, further including, if the queue includes at least a first thread,

removing the first thread from the queue; and

notifying the first thread that the first thread has acquired the lock.

7. (Original) The method of claim 1, further including, if

the selected action is trying to acquire the lock and

the state transition fails,

indicating that the lock was unable to be acquired.

8. (Original) The method of claim 1, further including, if

the state transition succeeds and

the selected action is trying to acquire the lock,

indicating the acquisition of the lock.

9. (Currently Amended) The method of claim 1, further including, if

the state transition succeeds,

the selected action is acquiring the lock, and

the speculatively determined ~~next~~second state represents the acquisition of the lock,

indicating the acquisition of the lock.

10. (Currently Amended) The method of claim 9, further including, if

the state transition succeeds,

the selected action is acquiring the lock, and

the speculatively determined ~~next~~second state does not represent the acquisition of the lock,

adding the thread to the end of a queue of threads waiting to acquire the lock;

waiting to receive notification that the thread may acquire the lock; and

indicating the acquisition of the lock.

11. (Currently Amended) The method of claim 1, further including, if

the state transition succeeds, and

the selected action is releasing the lock,

determining the number of threads in a queue to acquire the lock utilizing the speculatively determined ~~next~~second state of the lock.

12. (Original) The method of claim 11, further including, if the queue includes at least a first thread,

removing the first thread from the queue; and

notifying the first thread that the first thread has acquired the lock.

13. (Original) The method of claim 1, wherein the thread includes:

a unique thread identifier;

a next thread field to facilitate access to the next thread in a queue of threads waiting to acquire the lock; and

the thread is only capable of waiting for a single lock at a time.

14. (Original) The method of claim 1, wherein the action of acquiring the lock includes the inability to timeout or fail to acquire the lock.

15. (Currently Amended) The method of claim 1, wherein the lock's current state may change between

asynchronously querying the ~~current~~ first state of the lock; and

attempting to transition the lock from the queried ~~current~~ first state to the speculatively determined ~~next~~second state.

16. (Currently Amended) An apparatus comprising:

a processor;

a storage medium coupled to the processor, and having stored therein programming instructions to be operated by the processor to implement a lock acquirer

to acquire a lock, having a multi-part state value~~, including~~:

~~a flag value, a first thread value, and a last thread value~~ defined by a state machine associated with the lock;

wherein the lock acquirer performs an acquisition of the lock via

asynchronously querying and receiving ~~a current~~ a first state of the lock as a current state of the lock, by the lock acquirer, the lock being considered to be in any one of at least four states in any point in time, and each state is represented by the multi-part state value;

speculatively determining by the lock acquirer, ~~the~~a ~~next~~second state of the lock~~, where the next state is the state of the lock if the lock acquirer proceeds to perform the selected action and the lock acquirer is successful~~ based at least in part on the first state of the lock; and

attempting to perform, by the lock acquirer, the ~~selected action~~acquisition of the lock to transition the lock from the ~~current~~ first state to the speculatively determined ~~next~~second state, the attempting including determining if the first state remains the current state of the lock, and, if the first state remains the current state, performing the acquisition to transition the current state of the lock to the second state.

17. (Currently Amended) The apparatus of claim 16, wherein the lock acquirer is further capable of performing two general actions, including acquiring the lock, trying to acquire the lock; and

wherein, if the state transition fails and the general action is acquiring the lock, the lock acquirer is further capable of, repeating, until the state transaction succeeds:

asynchronously querying the ~~current~~ first state of the lock;

speculatively determining the ~~next~~second state of the lock; and

attempting to transition the lock from the queried ~~current~~ first state to the speculatively determined ~~next~~second state.

18. (Original) The apparatus of claim 17, wherein, if the state transition fails and the general action is trying to acquire the lock, the lock acquirer is further capable of,

indicating that the lock was unable to be acquired.

19. (Original) The apparatus of claim 18, wherein, if the state transition succeeds and the general action is trying to acquire the lock, the lock acquirer is further capable of,

indicating that the lock was acquired.

20. (Currently Amended) The apparatus of claim ~~16~~17, wherein, if

the state transition succeeds,

the general action is acquire the lock, and

the speculatively determined ~~next~~second state represents the acquisition of the lock,

the lock acquirer is further capable of,

indicating that the lock was acquired.

21. (Currently Amended) The apparatus of claim 20, wherein, if

the state transition fails,

the general action is acquire the lock, and

the speculatively determined ~~next~~second state does not represent the acquisition of the lock,

the lock acquirer is further capable of,

adding the thread to the end of a queue of threads waiting to acquire the lock;

waiting to receive notification that the thread may acquire the lock; and

indicating the acquisition of the lock.

22. (Original) The apparatus of claim 21, wherein the lock acquirer is unable to timeout of fail if the selected general action is acquiring the lock.

23. (Currently Amended) An apparatus comprising:

a processor;

a storage medium coupled to the processor, and having stored therein programming instructions to be operated by the processor to implement a lock releaser to release a lock, the lock having a multi-part state value defined by a state machine associated with the lock, ~~including:~~

~~a flag value, a first thread value, and a last thread value~~;

wherein the lock releaser, releases said hold on the lock via

asynchronously querying and receiving ~~a~~current a first state of the lock as a current state of the lock, by the lock releaser, the lock being considered to be in any one of at least four states in any point in time, and each state is represented by the multi-part state value;

speculatively determining by the lock releaser, ~~the~~a ~~next~~second state of the lock, ~~where the next state is the state of the lock if the lock releaser proceeds to perform the selected action and the lock releaser is successful~~ based at least in part on the first state of the lock; and

attempting to perform, by the lock releaser, ~~the selected action~~ the releasing of the lock to transition the lock from the ~~current~~first state to the speculatively determined ~~next~~second state, the attempting including determining if the first state remains the current state of the lock, and, if the first state remains the current state, performing the releasing to transition the current state of the lock to the second state.

24. (Currently Amended) The apparatus of claim 23, wherein, if the state transition fails, the lock releaser is further capable of, repeating, until the state transaction succeeds:

asynchronously querying the ~~current~~ first state of the lock;

speculatively determining the ~~next~~second state of the lock; and

attempting to transition the lock from the queried ~~current~~ first state to the speculatively determined ~~next~~second state.

25. (Currently Amended) The apparatus of claim 23, wherein, if the state transition succeeds, the lock releaser is further capable of determining the number of threads in a queue of threads waiting to acquire the lock utilizing the speculatively determined ~~next~~second state of the lock.

26. (Original) The apparatus of claim 25, wherein, if the queue includes at least a first thread, the lock releaser is further capable of:

removing the first thread from the queue; and

notifying the first thread that the first thread has acquired the lock.

27. (Original) The apparatus of claim 26, wherein the lock releaser is capable of removing the first thread from the queue utilizing a thread having:

a unique thread identifier; and

a next thread value to facilitate access to the next thread in the queue.

28. (Currently Amended) The apparatus of claim 23, wherein the lock is capable of changing state in between the time the lock releaser

asynchronously queries the ~~current~~ first state of the lock; and

attempts to transition the lock from the queried ~~current~~ first state to the speculatively determined ~~next~~second state.

29. (Currently Amended) An article comprising:

a storage medium; and

~~having~~ a plurality of ~~machine-accessible~~programming instructions stored on the storage medium and configured, when executed~~, wherein when the instructions are executed, the instructions provide for a plurality of threads executing~~ to coordinate access to a shared resource for a plurality of threads, including:

selecting by one of the threads an action to be performed a thread upon a lock utilized by the thread, wherein the action is selected from a group comprising:

acquiring the lock,

trying to acquire the lock, and

releasing the lock;

asynchronously querying ~~and~~ receiving a first state of the lock as a current state of the lock, by the thread, the lock ~~being~~ considered to be in any one of at least four states in any point in time, and each state is represented by a multi-part state value~~, each multi-part state value including a flag value, a first thread value, and a last thread value~~ defined by a state machine associated with the lock;

speculatively determining by the thread, ~~a~~the ~~next~~second state of the lock based at least in part on the first state of the lock and the selected action~~, where the next state is the state of the lock if the thread proceeds to perform the selected action and the thread is successful~~; and

attempting to perform by the thread, the selected action to transition the lock from the ~~current~~ first state to the speculatively determined ~~next~~second state, the attempting including determining if the first state remains the current state of the lock, and, if the first state remains the current state, performing the selected action to transition the current state of the lock to the second state.

30. (Currently Amended) The article of claim 29, further including instructions providing for, if the state transition fails and if the selected action was either acquiring or releasing the lock, repeating, until the state transition succeeds:

asynchronously querying the first~~current~~ state of the lock;

speculatively determining the ~~next~~second state of the lock;

attempting to transition the lock from the queried ~~current~~ first state to the speculatively determined ~~next~~second state.

31. (Currently Amended) The article of claim 30, further including instructions providing for, if

the state transition succeeds,

the selected action is acquiring the lock, and

the speculatively determined ~~next~~second state represents the acquisition of the lock,
indicating the acquisition of the lock.

32. (Currently Amended) The article of claim 31, further including instructions providing for, if
the state transition succeeds,
the selected action is acquiring the lock, and
the speculatively determined ~~next~~second state does not represent the acquisition of the lock,
adding the thread to the end of a queue of threads waiting to acquire the lock;
waiting to receive notification that the thread may acquire the lock; and
indicating the acquisition of the lock.

33. (Currently Amended) The article of claim 30, further including instructions providing for, if
the state transition succeeds, and
the selected action is releasing the lock,
determining the number of threads in a queue to acquire the lock utilizing the speculatively determined ~~next~~second state of the lock.

34. (Original) The article of claim 33, further including instructions providing for, if the queue includes at least a first thread,
removing the first thread from the queue; and
notifying the first thread that the first thread has acquired the lock.

35. (Original) The article of claim 29, further including instructions providing for, if
the selected action is trying to acquire the lock and
the state transition fails,
indicating that the lock was unable to be acquired.

36. (Original) The article of claim 29, further including instructions providing for, if

the state transition succeeds and

the selected action is trying to acquire the lock,

indicating the acquisition of the lock.

37. (Currently Amended) The article of claim 29, further including instructions providing for, if

the state transition succeeds,

the selected action is acquiring the lock, and

the speculatively determined ~~next~~second state represents the acquisition of the lock,

indicating the acquisition of the lock.

38. (Currently Amended) The article of claim 37, further including instructions providing for, if

the state transition succeeds,

the selected action is acquiring the lock, and

the speculatively determined ~~next~~second state does not represent the acquisition of the lock,

adding the thread to the end of a queue of threads waiting to acquire the lock;

waiting to receive notification that the thread may acquire the lock; and

indicating the acquisition of the lock.

39. (Currently Amended) The article of claim 29, further including instructions providing for, if

the state transition succeeds, and

the selected action is releasing the lock,

determining the number of threads in a queue to acquire the lock utilizing the speculatively determined ~~next~~second state of the lock.

40. (Original) The article of claim 39, further including instructions providing for, if the queue includes at least a first thread,

removing the first thread from the queue; and

notifying the first thread that the first thread has acquired the lock.

41. (Original) The article of claim 29, wherein the thread includes:

a unique thread identifier;

a next thread field to facilitate access to the next thread in a queue of threads waiting to acquire the lock; and

the thread is only capable of waiting for a single lock at a time.

42. (Original) The article of claim 29, wherein the action of acquiring the lock includes the inability to timeout or fail to acquire the lock.

43. (Currently Amended) The article of claim 29, wherein the lock's current state may change between

asynchronously querying the ~~current~~ first state of the lock; and

attempting to transition the lock from the queried ~~current~~ first state to the speculatively determined ~~next~~second state.

44.-55. (Cancelled)

56. (Currently Amended) The method of claim 1, wherein each of the ~~current~~ first and ~~next~~second states is a selected one of:

the lock is not held and there no threads waiting to access the shared resource,

the lock is held and there are no threads waiting to access the shared resource,

the lock is held and there is one thread waiting to access the shared resource, and

the lock is held and there are at least two threads waiting to access the shared resource.